

## Verificação e Validação

---

- O processo de verificação e validação
- O planeamento de verificação e validação
- Inspeções de software
- Análise estática automatizada
- Desenvolvimento de software Cleanroom

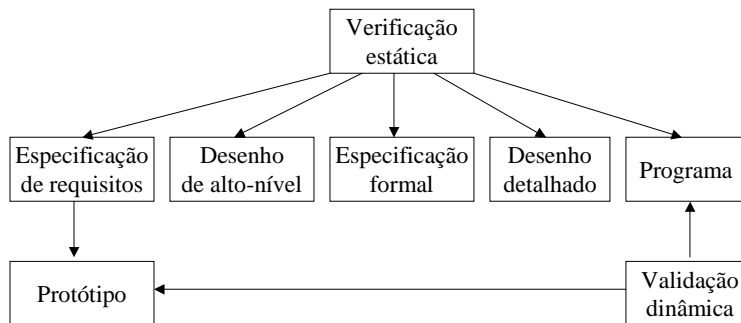
## Verificação vs. Validação

---

- Garantir que o SW satisfaz às necessidades do cliente
- Verificação:
  - “Estamos a construir o producto correctamente?”
  - O SW deve estar de acordo com a sua especificação
- Validação:
  - “Estamos a construir o producto desejado?”
- O processo de V & V
  - V & V devem ser aplicadas em cada estágio do processo de SW
  - Tem dois objectivos principais
    - » A descoberta de defeitos num sistema
    - » Avaliar se o sistema pode ser utilizado em uma situação operacional

## Verificação estática e dinâmica

- *Inspeção de Software* – Preocupa-se com a análise da representação estática de um sistema p/ a descoberta de problemas (verificação estática)
- *Teste de Software* - Preocupa-se em exercitar e observar o comportamento do produto (verificação dinâmica)



João Araújo 2000

Engenharia de Software

## Teste de programa

- Pode revelar a presença de erros, mas não a sua ausência
- É a principal forma de validação p/ requisitos não-funcionais
- Usado em conjunto com a verificação estática dá uma cobertura completa à V&V
- Tipos de teste
  - Testes de defeitos
    - » Testes desenhados p/ descobrir defeitos do sistema
  - Teste estatístico
    - » testes desenhados p/ refletir a frequência das entradas do utilizador. Usado p/ estimar a fiabilidade

João Araújo 2000

Engenharia de Software

## V& V: Objectivos

---

- O SW deve ser bom o bastante p/ o seu uso e o tipo de uso determinará o grau de confiança necessária
- V & V: a confiança depende do propósito do sistema, das expectativas do utilizador e do mercado
  - Função do software
    - » O nível de confiança depende de quão crítico é o SW p/ a organização
  - Expectativas do utilizador
    - » Utilizadores podem ter expectativas baixas de certos tipos de SW
  - Mercado
    - » Deve-se levar em conta os competidores e o preço que os clientes estão dispostos a pagar. Por o produto no mercado pode ser mais importante do que achar defeitos no programa

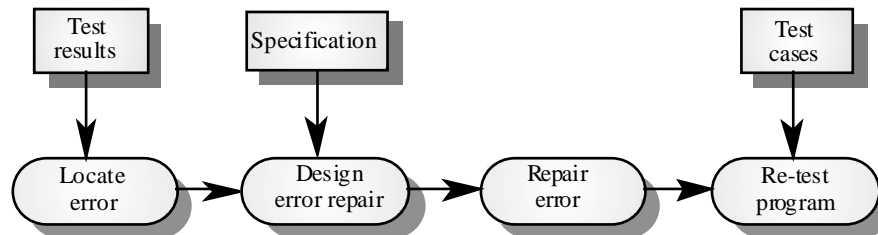
## Teste e debugging

---

- Teste de defeito e debugging são processos distintos
- V&V preocupa-se em estabelecer a existência de erros em um programa
- Debugging preocupa-se em localizar e reparar estes erros
- Debugging envolve a formulação de hipóteses sobre o comportamento de um programa e então testar estas hipóteses p/ descobrir o erro do sistema
- Novos testes são necessários após a correção (Teste por regressão)

## O processo de debugging

---

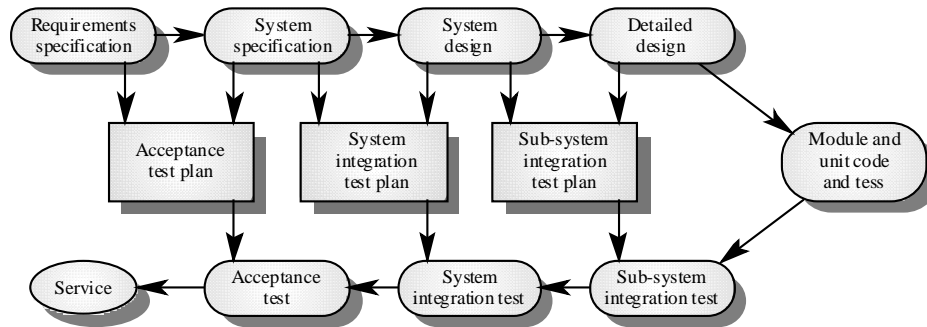


## V & V: Planeamento

---

- Planeamento é necessário p/ otimizar os processos de teste e inspeção. Deveria ser iniciado cedo no desenvolvimento
- O planeamento deveria incluir ambos verificação estática e teste
- O planeamento de testes define padrões p/ o processo de teste
- Estrutura:
  - O processo de teste
  - Traço (rastreamento) dos requisitos
  - Itens a serem testados
  - Escalonamento dos testes
  - Procedimentos de armazenamento de testes
  - Requisitos de hardware e software
  - Restrições

## O modelo V de desenvolvimento



João Araújo 2000

Engenharia de Software

## Inspeção de software

- Teste de programas requer um nº grande de testes a serem desenvolvidos, executados e examinados, o que consome tempo e é caro
- A inspeção consiste em examinar a representação fonte de um sistema (modelo, especificação, código) com objectivo de descobrir anomalias e defeitos
- Não requer a execução de um sistema. Pode ser aplicada a qualquer representação do sistema (requisitos, desenho, etc.)
- Sucesso da inspeção
  - Muitos defeitos podem ser descobertos numa inspeção simples. No teste um defeito pode mascarar um outro, portanto várias execuções são necessárias
  - Revisores estão alertas aos erros mais comuns

João Araújo 2000

Engenharia de Software

## Inspeções e testes

---

- Inspeções e testes são técnicas de verificação complementares utilizadas durante o processo de V&V
- Inspeções não conseguem validar os requisitos reais do utilizador
- Inspeções não podem verificar características não-funcionais tais como desempenho, utilização, etc.
- Inspeção de programa
  - O objectivo é a detecção de defeito
  - Defeitos podem ser lógicos, anomalias no código que podem indicar uma condição de erro (e.g. Variável não-inicializada) ou a não utilização dos padrões da organização

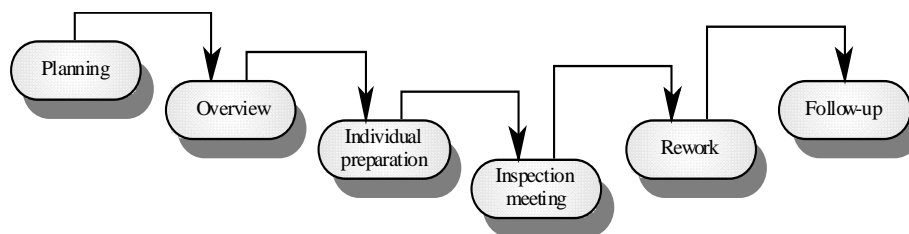
## Pré-condições p/ Inspeções

---

- Uma especificação precisa deve estar disponível
- Os membros da equipa devem estar familiarizados com os padrões da organização
- O código sintacticamente correcto deve estar disponível
- Uma lista de verificação de erros deve ser preparada (depende da linguagem)
- Procedimento de inspeção
  - Uma visão geral do sistema é apresentada à equipa de inspeção
  - O código e os documentos associados p/ a equipa de inspeção
  - A inspeção é realizada e os erros detectados são anotados
  - Modificações são realizadas p/ reparar erros descobertos
  - Re- inspeção pode ser ou não requerida
- Equipa de inspeção
  - Composta por pelo menos 4 membros
    - » Autor, inspector, moderador, leitor

## O processo de inspecção

---



## Listas de verificação p/ inspecção

---

- Listas de verificação dos erros mais comuns devem ser utilizadas p/ orientar a inspecção e dependem da linguagem de programação
- Quanto mais fraca a verificação de tipos maior é a lista
- Classes de falhas e exemplos de verificação
  - Dados: Inicialização de variáveis, nomes p/ constantes, limites de quadros, etc.
  - Controle: terminação de ciclo, condições correctas
  - Input/output: todas as variáveis de I/O estão sendo usadas correctamente?
  - Interface: ordem, tipo e número de parâmetros correctos
  - Gestão de memória: alocação e de-alocação
  - Gestão de exceções: todas as condições de erro possíveis

## Análise estática automatizada

---

- Analisadores estáticos são ferramentas de software p/ processar o código fonte de um programa, a fim de descobrir erros no programa
- É uma ajuda efectiva às inspecções, mas não uma substituição
- Utilização da análise estática
  - Particularmente vantajosa quando uma linguagem como C é usada (weak typing) onde muito erros não são detectados pelo compilador
  - Menos *cost-effective* p/ linguagens como Java que tem uma verificação de tipos forte e que portanto detecta muitos erros durante a compilação

## Estágios da análise estática

---

- *Análise do fluxo de controlo.* Verifica detecção de loop, código que nunca é executado, etc.
- *Análise da utilização dos dados.* Detecta variáveis usadas antes da inicialização, variáveis declaradas e nunca usadas, variáveis não declaradas, limites de quadros violados, etc.
- *Análise de interface.* Detecta erro de ordem, tipo e número de parâmetros, funções e procedimentos não chamados, não utilização de resultados de funções
- *Análise de fluxo da informação.* Identifica as dependências das variáveis de saída. Não detecta anomalias, mas chama a atenção p/ a revisão ou inspecção de código
- *Análise de passo.* Identifica passos através do programa e mostra os comandos executados em cada passo

138% more lint\_ex.c

```
#include <stdio.h>
printarray (Anarray)
int Anarray;
{
    printf("%d",Anarray);
}
main ()
{
    int Anarray[5]; int i; char c;
    printarray (Anarray, i, c);
    printarray (Anarray) ;
}
```

139% cc lint\_ex.c

140% lint lint\_ex.c

```
lint_ex.c(10): warning: c may be used before set
lint_ex.c(10): warning: i may be used before set
printarray: variable # of args. lint_ex.c(4) :: lint_ex.c(10)
printarray, arg. 1 used inconsistently lint_ex.c(4) ::
lint_ex.c(10)
printarray, arg. 1 used inconsistently lint_ex.c(4) ::
lint_ex.c(11)
printf returns value which is always ignored
```

Análise estática do LINT

## Desenvolvimento de SW Cleanroom

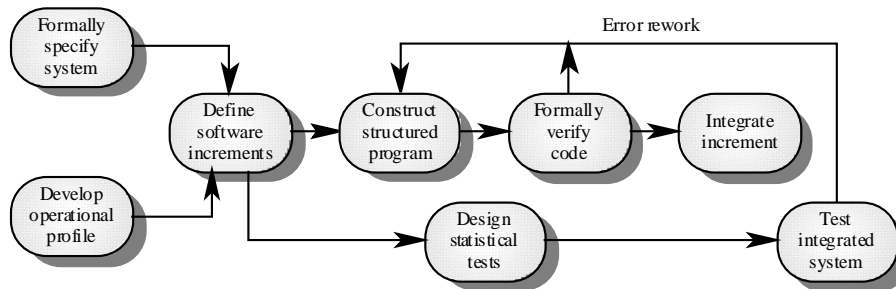
---

- A filosofia é antes evitar defeitos
- Características do processo
  - Desenvolvimento incremental
  - Especificação formal utilizando um modelo de transição de estados
  - Programação estruturada – refinamentos da especificação até criação do código
  - Verificação estática utilizando inspeções rigorosas
  - Teste estatístico do sistema

## O processo Cleanroom

- Equipas
  - *Equipa de especificação*. Responsável por desenvolver e manter a especificação do sistema
  - *Equipa de desenvolvimento*. Responsável por desenvolver e verificar o SW. Este não é documentado ou até mesmo compilado durante o processo
  - *Equipa de certificação*. Responsável por desenvolver um conjunto de testes estatísticos p/exercitar o SW depois do desenvolvimento. Modelos de crescimento de fiabilidade são usados p/ determinar quando a fiabilidade é aceitável
- Avaliação do processo Cleanroom
  - A utilização do Cleanroom tem resultado em SW com menos erros e não parece ser mais caro que o desenvolvimento convencional
  - Não parece que esta abordagem possa ser aplicada em qualquer ambiente

## O processo Cleanroom



# Desenvolvimento incremental

